

LU00f - LB02 Fokusthemen

Bei der LB2 im Modul 307 wollen wir uns mit den CRUD-Operationen, die Sie im Unterricht kennen gelernt haben, beschäftigen. Dabei steht das Acronym CRUD für die vier im Web gängigen Operationen:

- **C**rud = Create, Neuanlegen
- **c****R**ud = Read, Einzelansicht oder Listenansicht
- **cr****U**d = Update, Bearbeitung von einzelnen Datensätzen
- **crud** = Delete, Löschen von Datensätzen

Diese Operationen sollen, gemäss Input aus dem Unterricht, via HTML, JavaScript und der Middleware Express (Server in JS) umgesetzt werden. Ihre Lehrperson wird Ihrem Team eine der nachfolgenden LB2-Fokusthemen zuordnen.

1. Freunde

1.1 Ausgangslage

Sie erstellen eine einfache Webanwendung zur Verwaltung einer persönlichen Freundesliste. Die Anwendung besteht aus einer einzigen HTML-Seite mit eingebettetem JavaScript. Alle Daten werden in einer lokalen JSON-Datei (friends.json) gespeichert und von dort gelesen.

1.2 Aufgabenstellung

Erstellen Sie eine HTML-Seite index.html sowie eine Datendatei friends.json mit folgenden Funktionen:

1. Anzeigen (Read): Alle gespeicherten Freunde werden beim Laden der Seite als Liste angezeigt (Vorname, Nachname, Alter, Wohnort).
2. Hinzufügen (Create): Über ein Formular kann ein neuer Freund mit Vorname, Nachname, Alter und Wohnort erfasst und der Liste hinzugefügt werden.
3. Bearbeiten (Update): Ein bestehender Eintrag kann ausgewählt, im Formular bearbeitet und gespeichert werden.
4. Löschen (Delete): Jeder Listeneintrag kann über eine Schaltfläche entfernt werden.
5. Filtern: Es soll nach allen Kriterien gefiltert werden können.

1.3 Technische Anforderungen

- Die Datei friends.json enthält zu Beginn mindestens 3 Einträge als JSON-Array.
- Das Formular wird für die Auflistung, zur Eingabe und zur Bearbeitung verwendet (kein separates Formular).
- Pflichtfelder sind mit einer einfachen Validierung zu versehen (kein leerer Name).
- Die Anwendung soll ohne externe Frameworks auskommen (kein jQuery, kein Bootstrap).

2: Musiktitel

2.1 Ausgangslage

Sie erstellen eine einfache Webanwendung zur Verwaltung einer persönlichen Musiktitel-Sammlung. Die Anwendung besteht aus einer einzigen HTML-Seite mit eingebettetem JavaScript. Alle Daten werden in einer lokalen JSON-Datei (songs.json) gespeichert und von dort gelesen.

2.2 Aufgabenstellung

Erstellen Sie eine HTML-Seite index.html sowie eine Datendatei songs.json mit folgenden Funktionen:

1. Anzeigen (Read): Alle gespeicherten Musiktitel werden beim Laden der Seite als Liste angezeigt (Titel, Interpret, Album, Erscheinungsjahr).
2. Hinzufügen (Create): Über ein Formular kann ein neuer Musiktitel mit Titel, Interpret, Album und Erscheinungsjahr erfasst und der Liste hinzugefügt werden.
3. Bearbeiten (Update): Ein bestehender Eintrag kann ausgewählt, im Formular bearbeitet und gespeichert werden.
4. Löschen (Delete): Jeder Listeneintrag kann über eine Schaltfläche entfernt werden.
5. Filtern: Es soll nach allen Kriterien gefiltert werden können.

2.3 Technische Anforderungen

- Die Datei songs.json enthält zu Beginn mindestens 3 Einträge als JSON-Array.
- Das Formular wird zur Eingabe und zur Bearbeitung verwendet (kein separates Formular).
- Pflichtfelder (Titel und Interpret) sind mit einer einfachen Validierung zu versehen.
- Die Anwendung soll ohne externe Frameworks auskommen (kein jQuery, kein Bootstrap).

3: Subscriptions

3.1 Ausgangslage

Streaming-Dienste und digitale Abonnements (z. B. Netflix, Spotify, Disney+) gehören heute zum Alltag. Sie erstellen eine Webanwendung zur übersichtlichen Verwaltung solcher Abonnements. Die Anwendung besteht aus einer einzigen HTML-Seite mit eingebettetem JavaScript. Alle Daten werden in einer lokalen JSON-Datei (subscriptions.json) gespeichert und von dort gelesen.

3.2 Aufgabenstellung

Erstellen Sie eine HTML-Seite index.html sowie eine Datendatei subscriptions.json mit folgenden Funktionen:

1. Anzeigen (Read): Alle gespeicherten Abonnements werden beim Laden der Seite als Liste

angezeigt (Dienst, Kategorie, monatliche Kosten, Startdatum).

2. Hinzufügen (Create): Über ein Formular kann ein neues Abonnement mit Dienstname, Kategorie (z. B. Video, Musik, Software), monatlichen Kosten und Startdatum erfasst werden.
3. Bearbeiten (Update): Ein bestehender Eintrag kann ausgewählt, im Formular bearbeitet und gespeichert werden.
4. Löschen (Delete): Jeder Listeneintrag kann über eine Schaltfläche entfernt werden.
5. Filtern: Es soll nach allen Kriterien gefiltert werden können.

3.3 Technische Anforderungen

- Die Datei subscriptions.json enthält zu Beginn mindestens 3 Einträge als JSON-Array (z. B. Netflix, Spotify, Adobe Creative Cloud).
- Das Formular wird zur Eingabe und zur Bearbeitung verwendet (kein separates Formular).
- Pflichtfelder (Dienstname und Kosten) sind mit einer einfachen Validierung zu versehen; die Kosten dürfen keinen negativen Wert annehmen.
- Die Gesamtkosten aller aktiven Abonnements pro Monat werden unterhalb der Liste berechnet und angezeigt.
- Die Anwendung soll ohne externe Frameworks auskommen (kein jQuery, kein Bootstrap).

4. Fuhrpark

4.1 Ausgangslage

Unternehmen verwalten ihre Fahrzeuge häufig in einfachen Listen oder Tabellen. Sie erstellen eine Webanwendung zur digitalen Verwaltung eines Fuhrparks. Die Anwendung besteht aus einer einzigen HTML-Seite mit eingebettetem JavaScript. Alle Fahrzeugdaten werden in einer lokalen JSON-Datei (vehicles.json) gespeichert und von dort gelesen.

4.2 Aufgabenstellung

Erstellen Sie eine HTML-Seite index.html sowie eine Datendatei vehicles.json mit folgenden Funktionen:

1. Anzeigen (Read): Alle Fahrzeuge des Fuhrparks werden beim Laden der Seite als Liste angezeigt (Kennzeichen, Marke, Modell, Baujahr, Status).
2. Hinzufügen (Create): Über ein Formular kann ein neues Fahrzeug mit Kennzeichen, Marke, Modell, Baujahr und Status (verfügbar / in Betrieb / in Wartung) erfasst und der Liste hinzugefügt werden.
3. Bearbeiten (Update): Ein bestehender Fahrzeugeintrag kann ausgewählt, im Formular bearbeitet und gespeichert werden.
4. Löschen (Delete): Jeder Fahrzeugeintrag kann über eine Schaltfläche aus der Liste entfernt werden.
5. Filtern: Es soll nach allen Kriterien gefiltert werden können.

4.3 Technische Anforderungen

- Die Datei vehicles.json enthält zu Beginn mindestens 3 Einträge als JSON-Array (z. B. Lieferwagen, Personenwagen, Transporter).
- Das Formular wird zur Eingabe und zur Bearbeitung verwendet (kein separates Formular).
- Pflichtfelder (Kennzeichen und Marke) sind mit einer einfachen Validierung zu versehen; das Kennzeichen darf nicht doppelt vorkommen.
- Die Anzahl der Fahrzeuge je Status (verfügbar / in Betrieb / in Wartung) wird unterhalb der Liste angezeigt.
- Die Anwendung soll ohne externe Frameworks auskommen (kein jQuery, kein Bootstrap).

5. BZZ-Zimmerverwaltung

5.1 Ausgangslage

Sie erstellen eine einfache Webanwendung zur Verwaltung von Schulzimmern. Die Anwendung besteht aus einer einzigen HTML-Seite mit eingebettetem JavaScript. Alle Daten werden in einer lokalen JSON-Datei (rooms.json) gespeichert und von dort gelesen.

5.2 Aufgabenstellung

Erstellen Sie eine HTML-Seite index.html sowie eine Datendatei rooms.json mit folgenden Funktionen:

1. Anzeigen (Read): Alle gespeicherten Schulzimmer werden beim Laden der Seite als Liste angezeigt (Zimmernummer, Stockwerk, Kapazität, Ausstattung).
2. Hinzufügen (Create): Über ein Formular kann ein neues Schulzimmer mit Zimmernummer, Stockwerk, Kapazität und Ausstattung erfasst und der Liste hinzugefügt werden.
3. Bearbeiten (Update): Ein bestehender Eintrag kann ausgewählt, im Formular bearbeitet und gespeichert werden.
4. Löschen (Delete): Jeder Listeneintrag kann über eine Schaltfläche entfernt werden.
5. Filtern: Es soll nach allen Kriterien gefiltert werden können.

5.3 Technische Anforderungen

- Die Datei rooms.json enthält zu Beginn mindestens 3 Einträge als JSON-Array (z. B. H101, H131, H132).
- Das Formular wird zur Eingabe und zur Bearbeitung verwendet (kein separates Formular).
- Pflichtfelder (Zimmernummer, Kapazität und Ausstattung) sind mit einer einfachen Validierung zu versehen; das Kennzeichen darf nicht doppelt vorkommen.
- Die Anzahl der Zimmer je Status (verfügbar / nicht verfügbar) wird unterhalb der Liste angezeigt.
- Die Anwendung soll ohne externe Frameworks auskommen (kein jQuery, kein Bootstrap).

6. EFZ/BMS/ABU-Zeugnis

6.1 Ausgangslage

Sie erstellen eine einfache Webanwendung zur Verwaltung von EFZ-Notenausweisen für Mediamatiker/innen. Die Anwendung besteht aus einer einzigen HTML-Seite mit eingebettetem JavaScript. Alle Daten werden in einer lokalen JSON-Datei (zeugnisse.json) gespeichert und von dort gelesen.

6.2 Aufgabenstellung

Erstellen Sie eine HTML-Seite index.html sowie eine Datendatei zeugnisse.json mit folgenden Funktionen:

1. Anzeigen (Read): Alle gespeicherten Notenausweise werden beim Laden der Seite als Liste angezeigt (Name, Vorname, Lehrjahr, Gesamtnote, Bestanden/Nicht bestanden).
2. Hinzufügen (Create): Über ein Formular kann ein neuer Notenausweis mit Name, Vorname, Lehrjahr sowie den vier Qualifikationsbereichsnoten (VPA, Mediamatikkompetenzen, Allgemeinbildung, Betrieb) erfasst werden.
3. Bearbeiten (Update): Ein bestehender Eintrag kann ausgewählt, im Formular bearbeitet und gespeichert werden.
4. Löschen (Delete): Jeder Listeneintrag kann über eine Schaltfläche entfernt werden.
5. **Bonus:** Gesamtnote berechnen: Die Gesamtnote wird automatisch aus den vier Qualifikationsbereichsnoten gemäss der offiziellen Gewichtung berechnet (VPA 30%, Mediamatikkompetenzen 30%, Allgemeinbildung 10%, Betrieb 20%, Erweiterte Grundkompetenzen 10%) und in der Liste angezeigt.
6. Filtern: Es soll nach allen vier Kriterien gefiltert werden können.

6.3 Technischen Anforderungen

- Die Datei zeugnisse.json enthält zu Beginn mindestens 3 Einträge als JSON-Array.
- Jeder Eintrag enthält mindestens: name, vorname, lehrjahr, noteVPA, noteMediamatik, noteAGB, noteBetrieb (alle als Zahl, halbe Noten erlaubt).
- Das Formular wird zur Eingabe und zur Bearbeitung verwendet (kein separates Formular).
- Pflichtfelder sind mit einer einfachen Validierung zu versehen (kein leerer Name, Noten zwischen 1.0 und 6.0).
- **Bonus:** Ein Eintrag wird in der Liste farblich hervorgehoben, wenn er nicht bestanden ist (Gesamtnote < 4.0).
- Die Anwendung soll ohne externe Frameworks auskommen (kein jQuery, kein Bootstrap).

7. Kundenverwaltung

7.1 Ausgangslage

Sie erstellen eine einfache Webanwendung zur Verwaltung von Kunden. Die Anwendung besteht aus

einer einzigen HTML-Seite mit eingebettetem JavaScript. Alle Daten werden in einer lokalen JSON-Datei (kunden.json) gespeichert und von dort gelesen.

7.2 Aufgabenstellung

Erstellen Sie eine HTML-Seite index.html sowie eine Datendatei kunden.json mit folgenden Funktionen:

- Anzeigen (Read): Alle gespeicherten Kunden werden beim Laden der Seite als Liste angezeigt (Vorname, Nachname, E-Mail, Telefon, Ort).
- Hinzufügen (Create): Über ein Formular kann ein neuer Kunde mit Vorname, Nachname, E-Mail, Telefon und Ort erfasst und der Liste hinzugefügt werden.
- Bearbeiten (Update): Ein bestehender Eintrag kann ausgewählt, im Formular bearbeitet und gespeichert werden.
- Löschen (Delete): Jeder Listeneintrag kann über eine Schaltfläche entfernt werden.
- Filtern: Es soll nach allen Kriterien gefiltert werden können.

7.3 Technischen Anforderungen

Technische Anforderungen

- Die Datei kunden.json enthält zu Beginn mindestens 3 Einträge als JSON-Array.
- Das Formular wird zur Eingabe und zur Bearbeitung verwendet (kein separates Formular).
- Pflichtfelder sind mit einer einfachen Validierung zu versehen (kein leerer Name, gültige E-Mail-Adresse).
- Die Anwendung soll ohne externe Frameworks auskommen (kein jQuery, kein Bootstrap).

8. Lehrbetriebe

8.1 Ausgangslage

Sie erstellen eine einfache Webanwendung zur Verwaltung von Lehrbetrieben. Die Anwendung besteht aus einer einzigen HTML-Seite mit eingebettetem JavaScript. Alle Daten werden in einer lokalen JSON-Datei (lehrbetriebe.json) gespeichert und von dort gelesen.

8.2 Aufgabenstellung

Erstellen Sie eine HTML-Seite index.html sowie eine Datendatei lehrbetriebe.json mit folgenden Funktionen:

1. Anzeigen (Read): Alle gespeicherten Lehrbetriebe werden beim Laden der Seite als Liste angezeigt (Firmenname, Branche, Kontaktperson, Telefon, Ort).
2. Hinzufügen (Create): Über ein Formular kann ein neuer Lehrbetrieb mit Firmenname, Branche, Kontaktperson, Telefon und Ort erfasst und der Liste hinzugefügt werden.
3. Bearbeiten (Update): Ein bestehender Eintrag kann ausgewählt, im Formular bearbeitet und

gespeichert werden.

4. Löschen (Delete): Jeder Listeneintrag kann über eine Schaltfläche entfernt werden.
5. Filtern: Es soll nach allen Kriterien gefiltert werden können.

8.3 Technischen Anforderungen

- Die Datei lehrbetriebe.json enthält zu Beginn mindestens 3 Einträge als JSON-Array.
- Das Formular wird zur Eingabe und zur Bearbeitung verwendet (kein separates Formular).
- Pflichtfelder sind mit einer einfachen Validierung zu versehen (kein leerer Firmenname, kein leere Kontaktperson).
- Die Anwendung soll ohne externe Frameworks auskommen (kein jQuery, kein Bootstrap).

9. Staatsoberhäupter & Prime-Minister

9.1 Ausgangslage

Sie erstellen eine einfache Webanwendung zur Verwaltung von Staatsoberhäuptern und Präsidenten. Die Anwendung besteht aus einer einzigen HTML-Seite mit eingebettetem JavaScript. Alle Daten werden in einer lokalen JSON-Datei (staatsoberhaupter.json) gespeichert und von dort gelesen.

9.2 Aufgabenstellung

Erstellen Sie eine HTML-Seite index.html sowie eine Datendatei staatsoberhaupter.json mit folgenden Funktionen:

1. Anzeigen (Read): Alle gespeicherten Staatsoberhäupter werden beim Laden der Seite als Liste angezeigt (Vorname, Nachname, Land, Titel, Amtsantritt).
2. Hinzufügen (Create): Über ein Formular kann ein neues Staatsoberhaupt mit Vorname, Nachname, Land, Titel (z. B. Präsident, König, Bundeskanzler) und Amtsantritt (Jahr) erfasst und der Liste hinzugefügt werden.
3. Bearbeiten (Update): Ein bestehender Eintrag kann ausgewählt, im Formular bearbeitet und gespeichert werden.
4. Löschen (Delete): Jeder Listeneintrag kann über eine Schaltfläche entfernt werden.
5. **Bonus** Amtsdauer berechnen: Die bisherige Amtsdauer in Jahren wird automatisch aus dem Amtsantritt berechnet und in der Liste angezeigt.
6. Filtern: Es soll nach allen Kriterien gefiltert werden können.

9.3 Technische Anforderungen

- Die Datei staatsoberhaupter.json enthält zu Beginn mindestens 3 Einträge als JSON-Array.
- Jeder Eintrag enthält mindestens: vorname, nachname, land, titel, amtsantritt (als vierstellige Jahreszahl).
- Das Formular wird zur Eingabe und zur Bearbeitung verwendet (kein separates Formular).
- Pflichtfelder sind mit einer einfachen Validierung zu versehen (kein leerer Name, kein leeres Land, Amtsantritt muss eine gültige vierstellige Jahreszahl sein).
- Die Anwendung soll ohne externe Frameworks auskommen (kein jQuery, kein Bootstrap).

10. Gebäudeverwaltung

10.1 Ausgangslage

Sie erstellen eine einfache Webanwendung zur Verwaltung der Gebäude und Schulstandorte einer Schule. Die Anwendung besteht aus einer einzigen HTML-Seite mit eingebettetem JavaScript. Alle Daten werden in einer lokalen JSON-Datei (gebäude.json) gespeichert und von dort gelesen.

10.2 Aufgabenstellung

Erstellen Sie eine HTML-Seite index.html sowie eine Datendatei gebäude.json mit folgenden Funktionen:

1. Anzeigen (Read): Alle gespeicherten Gebäude werden beim Laden der Seite als Liste angezeigt (Bezeichnung, Adresse, Anzahl Stockwerke, Anzahl Schulzimmer, Baujahr).
2. Hinzufügen (Create): Über ein Formular kann ein neues Gebäude mit Bezeichnung, Adresse, Anzahl Stockwerke, Anzahl Schulzimmer und Baujahr erfasst und der Liste hinzugefügt werden.
3. Bearbeiten (Update): Ein bestehender Eintrag kann ausgewählt, im Formular bearbeitet und gespeichert werden.
4. Löschen (Delete): Jeder Listeneintrag kann über eine Schaltfläche entfernt werden.
5. **Bonus:** Gebäudealter berechnen: Das Alter des Gebäudes in Jahren wird automatisch aus dem Baujahr berechnet und in der Liste angezeigt.
6. Filtern: Es soll nach allen Kriterien gefiltert werden können.

10.3 Technische Anforderungen

- Die Datei gebäude.json enthält zu Beginn mindestens 3 Einträge als JSON-Array.
- Jeder Eintrag enthält mindestens: bezeichnung, adresse, stockwerke, schulzimmer, baujahr (als vierstellige Jahreszahl).
- Das Formular wird zur Eingabe und zur Bearbeitung verwendet (kein separates Formular).
- Pflichtfelder sind mit einer einfachen Validierung zu versehen (keine leere Bezeichnung, keine leere Adresse, Baujahr als gültige vierstellige Jahreszahl).
- Die Anwendung soll ohne externe Frameworks auskommen (kein jQuery, kein Bootstrap).

11. Ferienverwaltung

11.1 Ausgangslage

Sie erstellen eine einfache Webanwendung zur Verwaltung persönlicher Ferienreisen. Die Anwendung ermöglicht es, vergangene und geplante Reisen festzuhalten – wohin man gereist ist, wann, wie lange und wie viel die Reise gekostet hat. Die Anwendung besteht aus einer einzigen HTML-Seite mit eingebettetem JavaScript. Alle Daten werden in einer lokalen JSON-Datei (reisen.json) gespeichert und von dort gelesen.

11.2 Aufgabenstellung

Erstellen Sie eine HTML-Seite `index.html` sowie eine Datendatei `reisen.json` mit folgenden Funktionen:

1. Anzeigen (Read): Alle gespeicherten Reisen werden beim Laden der Seite als Liste angezeigt (Reiseziel, Land, Abreisedatum, Rückreisedatum, Transportmittel, Kosten in CHF).
2. Hinzufügen (Create): Über ein Formular kann eine neue Reise mit Reiseziel, Land, Abreisedatum, Rückreisedatum, Transportmittel (z. B. Flugzeug, Zug, Auto, Schiff) und Kosten in CHF erfasst werden.
3. Bearbeiten (Update): Ein bestehender Eintrag kann ausgewählt, im Formular bearbeitet und gespeichert werden.
4. Löschen (Delete): Jeder Listeneintrag kann über eine Schaltfläche entfernt werden.
5. **Bonus** Reisedauer berechnen: Die Reisedauer in Tagen wird automatisch aus Abreise- und Rückreisedatum berechnet und in der Liste angezeigt.
6. **Bonus** Gesamtkosten anzeigen: Unterhalb der Liste wird die Summe aller Reisekosten in CHF automatisch berechnet und angezeigt.
7. Filtern: Es soll nach allen Kriterien gefiltert werden können.

11.3 Technische Anforderungen

Technische Anforderungen

- Die Datei `reisen.json` enthält zu Beginn mindestens 3 Einträge als JSON-Array.
- Jeder Eintrag enthält mindestens: `reiseziel`, `land`, `abreise` (YYYY-MM-DD), `rueckreise` (YYYY-MM-DD), `transportmittel`, `kosten` (Zahl).
- Das Formular wird zur Eingabe und zur Bearbeitung verwendet (kein separates Formular).
- Pflichtfelder sind mit einer einfachen Validierung zu versehen (kein leeres Reiseziel, kein leeres Land, Rückreisedatum darf nicht vor dem Abreisedatum liegen).
- Die Anwendung soll ohne externe Frameworks auskommen (kein jQuery, kein Bootstrap).

12. Strafzettel-Management

12.1 Ausgangslage

Sie entwickeln eine einfache Webanwendung zur Verwaltung von Strafzetteln (Verkehrsbussen). Die Anwendung ermöglicht es, ausgestellte Strafzettel zu erfassen, einzusehen, zu bearbeiten und zu löschen.

Zu jedem Strafzettel werden Angaben zum Fahrzeughalter, dem Kennzeichen, der Verstossart, dem Ort sowie der Busse in CHF gespeichert. Die Anwendung besteht aus einer einzigen HTML-Seite mit eingebettetem JavaScript. Alle Daten werden in einer lokalen JSON-Datei (`strafzettel.json`) gespeichert und von dort gelesen.

12.2 Aufgabenstellung

Erstellen Sie eine HTML-Seite `index.html` sowie eine Datendatei `strafzettel.json` mit folgenden

Funktionen:

1. Anzeigen (Read): Alle gespeicherten Strafzettel werden beim Laden der Seite als Liste angezeigt (Name des Halters, Kennzeichen, Verstoss, Ort, Datum, Bussbetrag in CHF).
2. Hinzufügen (Create): Über ein Formular kann ein neuer Strafzettel mit Name des Fahrzeughalters, Kennzeichen, Verstossart (z. B. Falschparken, Geschwindigkeitsübertretung, Rotlichtmissachtung, Handynutzung am Steuer), Ort, Datum und Bussbetrag in CHF erfasst werden.
3. Bearbeiten (Update): Ein bestehender Eintrag kann ausgewählt, im Formular bearbeitet und gespeichert werden.
4. Löschen (Delete): Jeder Listeneintrag kann über eine Schaltfläche entfernt werden.
5. **Bonus** Bussenkategorie anzeigen: Anhand des Bussbetrags wird automatisch eine Kategorie berechnet und angezeigt: «Gering» (unter 100 CHF), «Mittel» (100–299 CHF), «Hoch» (ab 300 CHF).
6. **Bonus** Gesamtbussen anzeigen: Unterhalb der Liste wird die Summe aller Bussbeträge in CHF automatisch berechnet und angezeigt.
7. Filtern: Es soll nach allen Kriterien gefiltert werden können.

12.3 Technische Anforderungen

- Die Datei strafzettel.json enthält zu Beginn mindestens 3 Einträge als JSON-Array.
- Jeder Eintrag enthält mindestens: name, kennzeichen, verstoss, ort, datum (YYYY-MM-DD), betrag (Zahl).
- Das Formular wird zur Eingabe und zur Bearbeitung verwendet (kein separates Formular).
- Pflichtfelder sind mit einer einfachen Validierung zu versehen: kein leeres Namensfeld, kein leeres Kennzeichen, Datum darf nicht in der Zukunft liegen, Betrag muss eine positive Zahl sein.
- Die Anwendung soll ohne externe Frameworks auskommen (kein jQuery, kein Bootstrap).

13. Equipment-Verwaltung

13.1 Ausgangslage

Sie entwickeln eine einfache Webanwendung zur Verwaltung von Equipment (z. B. Kameras, Stative, Blitzgeräte, Notebooks, Mikrofone). Die Anwendung ermöglicht es, vorhandene Geräte zu erfassen, einzusehen, zu bearbeiten und zu löschen.

Zu jedem Gerät werden Angaben zur Bezeichnung, Kategorie, Seriennummer, Standort, Anschaffungsdatum und Wert in CHF gespeichert. Die Anwendung besteht aus einer einzigen HTML-Seite mit eingebettetem JavaScript. Alle Daten werden in einer lokalen JSON-Datei (equipment.json) gespeichert und von dort gelesen.

13.2 Aufgabenstellung

Erstellen Sie eine HTML-Seite index.html sowie eine Datendatei equipment.json mit folgenden Funktionen:

1. Anzeigen (Read): Alle gespeicherten Geräte werden beim Laden der Seite als Liste angezeigt (Bezeichnung, Kategorie, Seriennummer, Standort, Anschaffungsdatum, Wert in CHF).
2. Hinzufügen (Create): Über ein Formular kann ein neues Gerät mit Bezeichnung, Kategorie (z. B. Kamera, Stativ, Blitz, Notebook, Mikrofon, Sonstiges), Seriennummer, Standort, Anschaffungsdatum und Wert in CHF erfasst werden.
3. Bearbeiten (Update): Ein bestehender Eintrag kann ausgewählt, im Formular bearbeitet und gespeichert werden.
4. Löschen (Delete): Jeder Listeneintrag kann über eine Schaltfläche entfernt werden.
5. **Bonus** Alter des Geräts berechnen: Das Alter des Geräts in Jahren wird automatisch aus dem Anschaffungsdatum berechnet und in der Liste angezeigt.
6. **Bonus** Gesamtwert anzeigen: Unterhalb der Liste wird die Summe aller Gerätewerte in CHF automatisch berechnet und angezeigt.
7. Filtern nach Kategorie: Über ein Dropdown kann die Liste nach Kategorie gefiltert werden, sodass nur Geräte der gewählten Kategorie angezeigt werden.

13.3 Technische Anforderungen

- Die Datei equipment.json enthält zu Beginn mindestens 3 Einträge als JSON-Array.
- Jeder Eintrag enthält mindestens: bezeichnung, kategorie, seriennummer, standort, anschaffung (YYYY-MM-DD), wert (Zahl).
- Das Formular wird zur Eingabe und zur Bearbeitung verwendet (kein separates Formular).
- Pflichtfelder sind mit einer einfachen Validierung zu versehen: keine leere Bezeichnung, keine leere Seriennummer, Anschaffungsdatum darf nicht in der Zukunft liegen, Wert muss eine positive Zahl sein.
- Die Anwendung soll ohne externe Frameworks auskommen (kein jQuery, kein Bootstrap).

✖ Volkan Demir

From:

<https://wiki.bzz.ch/> - **BZZ - Modulwiki**

Permanent link:

<https://wiki.bzz.ch/de/modul/m307/learningunits/lu00/07?rev=1780580794>

Last update: **2026/06/04 15:46**

